Лабораторная работа № 2 по курсу : Операционные системы

Выполнил студент группы М8О-201Б-21 Кварацхелия Александр.

Условие

Ознакомиться с сигналами операционной системы UNIX/LINUX, используя утилиту strace, проанализировать результаты, сопоставить их с кодом программы.

Метод решения

Использовать свободно распространяемую утилиту strace следующим образом: strace lab2

Код программы

libLab5.h

```
typedef double(*TDerivative)(double, double);
typedef double(*TIntegral)(double, double, double);

double Derivative(double a, double deltaX);
double Integrate(double a, double b, double epsilon);
```

lib1.c

```
#include <math.h>

double Derivative(double a, double deltaX){
    return (cos(a + deltaX) - cos(a)) / deltaX;
}

double Integrate(double a, double b, double epsilon)
{
    int steps = fabs(b - a) / epsilon;
    double point = a;
    double result = 0;

    for (int i = 0; i < steps; ++i)
    {
        result += sin(point) * epsilon;
        point += epsilon;
```

```
    }

    return result;
}
```

lib2.c

```
#include <stdio.h>
#include <math.h>

double Integrate(double a, double b, double epsilon)
{
    int steps = fabs(b - a) / epsilon;
    double point = a;
    double result = 0;

    for (int i = 0; i < steps; ++i)
    {
        result += sin(point + epsilon / 2) * epsilon;
        point += epsilon;
    }

    return result;
}

double Derivative(double a, double deltaX){
    return (cos(a + deltaX) - cos(a - deltaX)) / (2 * deltaX);
}
```

program1.c

```
#include <stdio.h>
#include <libLab5.h>

int main()
{
    printf("1 - integrate, 2 - find derivative, -1 - quit\n");

    int command;

    while (scanf("%d", &command) != EOF)
    {
        if (command == 1)
        {
```

```
            printf("Pass arguments: \"a b epsilon\"\n");

            double a, b, epsilon;
            scanf("%lf %lf %lf", &a, &b, &epsilon);

            printf("%lf\n", Integrate(a, b, epsilon));
        }
        else if (command == 2)
        {
            printf("Pass arguments: \"a deltaX\"\n");

            double a, deltaX;
            scanf("%lf %lf", &a, &deltaX);

            printf("%lf\n", Derivative(a, deltaX));
        }
        else if (command == -1){
            break;
        }
        else{
            return 0;
        }
    }

    return 0;
}
```

program2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#include "libLab5.h"

int main()
{
    char* libNames[] = {"./libdynamic1.so", "./libdynamic2.so"};
    int curLib = 0;
    int numOfLibs = 2;

    void* libCtrl;
    libCtrl  = dlopen(libNames[curLib], RTLD_NOW);
```

```
if (!libCtrl)
{
    perror("Openning dynamic lib error )");
    exit(EXIT_FAILURE);
}

TDerivative derivative;
TIntegral integral;

#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wpedantic"
    derivative = (TDerivative)dlsym(libCtrl, "Derivative");
    integral = (TIntegral)dlsym(libCtrl, "Integrate");
#pragma GCC diagnostic pop

printf("0 - switch library (default - derivative), 1 - find derivative , 2

int command;

while(scanf("%d", &command) != EOF)
{
    if (command == 0)
    {
        curLib = (curLib + 1) % numOfLibs;

        if (dlclose(libCtrl ) != 0)
        {
            perror("closing dynamic lib error");
            exit(EXIT_FAILURE);
        }

        if (!(libCtrl = dlopen(libNames[curLib], RTLD_LAZY)))
        {
            perror("closing dynamic lib error");
            exit(EXIT_FAILURE);
        }

        #pragma GCC diagnostic push
        #pragma GCC diagnostic ignored "-Wpedantic"
            derivative = (TDerivative)dlsym(libCtrl, "Derivative");
            integral = (TIntegral)dlsym(libCtrl, "Integrate");
        #pragma GCC diagnostic pop
```

```
        }
        else if (command == 1)
        {
            printf("Pass arguments: \"a deltaX\"\n");

            double a, deltaX;
            scanf("%lf %lf", &a, &deltaX);

            double result = (derivative)(a, deltaX);
            printf("%lf\n", result);
        }
        else if (command == 2)
        {
            printf("Pass arguments: \"a b epsilon\"\n");

            double a, b, epsilon;
            scanf("%lf %lf %lf", &a, &b, &epsilon);

            double result = (integral)(a, b, epsilon);
            printf("%lf\n", result);
        }
        else if (command == -1){
            break;
        }
        else{
            return 0;
        }
    }

    return 0;
}
```

Выводы

Вызов *fork* дублирует породивший его процесс со всеми его переменными, файловыми дескрипторами, приоритетами процесса, рабочий и корневой каталоги, и сегментами выделенной памяти.

Ребёнок **не** наследует:

- идентификатора процесса (PID, PPID);
- израсходованного времени ЦП (оно обнуляется);
- сигналов процесса-родителя, требующих ответа;

- блокированных файлов (record locking).

В процессе выполнения лабораторной работы были приобретены навыки практического применения создания, обработки и отслеживания их состояния. Для выполнения данного варианта задания создание потоков как таковых не требуется, так как всю работу выполняет системный вызов «exec».